A particle swarm optimization for the no-wait flow shop problem with due date constraints

Hamed Samarghandi
Department of Finance and Management Science, Edwards School of Business,
University of Saskatchewan, Saskatoon, Saskatchewan, Canada, S7N 5A7
samarghandi@edwards.usask.ca

Abstract

This paper considers the no-wait flow shop scheduling problem with due date constraints. In the no-wait flow shop problem, waiting time is not allowed between successive operations of jobs. Moreover, a due date is associated with the completion of each job. The considered objective function is makespan. This problem is proved to be strongly NP-Hard. In this paper a particle swarm optimization (PSO) is developed to deal with the problem. Moreover, the effect of some dispatching rules for generating initial solutions are studied. A Taguchi-based design of experience approach has been followed to determine the effect of the different values of the parameters on the performance of the algorithm. To evaluate the performance of the proposed PSO, a large number of benchmark problems are selected from the literature and solved with different due date and penalty settings. Computational results confirm that the proposed PSO is efficient and competitive; the developed framework is able to improve many of the best-known solutions of the test problems available in the literature.

Keywords: Flow Shop Scheduling; No-wait; Due Date Constraints; Makespan; Particle Swarm Optimization

1 Introduction

The no-wait flow shop problem is a special case of the classical flow shop problem, in which there should be no waiting time between successive operations of jobs. In other words, once processing is started, no interruption is permitted between operations of the same job. Moreover, in this paper, completion of each job is associated with a due date. In other words, jobs must be completed before their due dates. Due

date constraints are among the most applicable constraints in scheduling and sequencing literature because real-world jobs are usually accompanied by a deadline for completion (Hunsucker and Shah 1992). In this paper it is assumed that all the jobs are ready at time zero (all release dates are zero) and the considered performance measure is makespan.

Following the three-field notation of the scheduling problems, the problem can be designated as $F|no-wait,d_i|C_{\max}$. It can be noted that the objective function of the considered problem is makespan and since the objective function does not provide any hints about the due date constraints, these constraints should be explicitly mentioned in the field. No-wait scheduling problem was first studied by Piehler (1960). King and Spachis (1980) proved that the no-wait flow shop problem with makespan performance measure ($F|no-wait|C_{\max}$) can be transformed to the Asymmetric Travelling Salesperson Problem (ATSP). Röck (1984) proved that ($F|no-wait|C_{\max}$) is NP-Hard. Since $F|no-wait,d_i|C_{\max}$ is a generalization of $F|no-wait|C_{\max}$, it can be inferred that $F|no-wait,d_i|C_{\max}$ is also strongly NP-Hard.

Industrial applications mentioned in the literature for $F|no-wait,d_i|C_{\max}$ include chemical industries (Rajendran 1994), food industries (Hall and Sriskandarajah 1996), steel production (Wismer 1972), pharmaceutical industries (Raaymakers and Hoogeveen 2000), and production of concrete products (Grabowski and Pempera 2000). Hall and Sriskandarajah (1996) provide a comprehensive review of the applications of the problem.

If a job is finished after its completion deadline, the company will face considerable penalties for compensation. Companies always look for methods to reduce waste and improve efficiency; therefore, reducing the number of late days or the number of tardy jobs is important when a company tries to avoid penalties (Pinedo 2002). Furthermore, a contractor will tremendously damage its reputation as a reliable firm if it frequently delivers jobs after their due dates are passed (even if the number of late days is relatively

small). Trust between companies will be also damaged if late jobs are not frequent, but a few jobs are delivered considerably past their due dates. Note that on-time delivery of the jobs can be only one of the goals of a company. Companies can be interested in optimizing other criteria such as makespan, while reducing the number of late days or tardy jobs. Hence, $F \mid no-wait, d_i \mid C_{\max}$ is not only an applicable problem with many real-world applications, but it is proved to be NP-Hard.

Hunsucker and Shah (1992) studied the problem of scheduling a constrained flow shop with multiple processors and due dates. The main purpose of their study was to evaluate the performance of six priority rules under different congestion levels with mean tardiness and number of tardy jobs as the performance measure. The performance of 10 different priority dispatching rules in a multi-processor flow shop for mean and maximum tardiness was studied in Brah (1996). Sarper (1995) studied the problem of minimization of the sum of absolute deviations of job completion times from a common due date for the two-machine flow shop problem. Gupta *et al.* (2000) studied a two-machine flow shop problem in which all the jobs have a common due date and developed an enumerative algorithm with branch and bound components to minimize either the sum or the maximum earliness and tardiness. Gowrishankar *et al.* (2001) considered the problem of minimizing the variance of completion of jobs and minimizing the sum of squares of deviations of job completion times from a common due date. Błażewicz *et al.* (2005) studied a two-machine flow shop problem with a common due date and developed a dynamic programming approach to minimize the total weighted late work criterion.

More recently, Błażewicz *et al.* (2008) developed a number of metaheuristic algorithms for the two-machine flow shop problem with a common due date and the weighted late work performance measure. Panwalkar and Koulamas (2012) considered the problem of minimizing a common due date and the number of tardy jobs in a two-machine flow shop syntax in which the first operation of each job is smaller than the second operation and developed an $O(n^2)$ algorithm for this problem. Tang *et al.* (2011) considered a flow shop environment with due date constraints and developed a hybrid of particle swarm optimization and

knowledge evolution algorithm. Ebrahimi *et al.* (2014) studied a hybrid flow shop environment with sequence dependent setup times and uncertain due dates with minimization of makespan and total tardiness as the objective functions and developed two metaheuristic algorithms to deal with this problem. Tari and Olfat (2014) considered the tardiness flow shop problem with intermediate due dates. They developed a number of heuristic priority dispatching rules for this problem and compared their performance using computational results. Dhingra and Chandna (2010) developed a number of metaheuristic methods to deal with the flow shop problem with sequence dependent setup times and due date constraints.

It can be noted that although the flow shop problem and its variants in the presence of due date constraints have received a lot of attention in the literature, the problem of $F\,|\,no-wait,d_i\,|\,C_{\max}$ has not yet been studied. Table 1 summarizes the most relevant research efforts in the area of no-wait flow shop scheduling.

In this paper, a Particle Swarm Optimization (PSO) is developed to deal with $F\,|\,no-wait,d_i\,|\,C_{\max}$. Moreover, an algorithm is developed to efficiently create a timetable from a given sequence. The timetabling algorithm is further coupled with the developed PSO to explore the feasible region of the problem. Although $F\,|\,no-wait,d_i\,|\,C_{\max}$ has numerous practical applications, it has received little attention in the literature. In this paper, different due date settings are considered, and computational results are compared with the results of the most competitive methods in the literature for $F\,|\,no-wait\,|\,C_{\max}$.

The contribution of this paper is threefold. First, $F\,|\,no-wait,d_i\,|\,C_{\max}$ is studied in this paper for the first time; a mathematical model is developed for this problem and a number of small-instance test problems are solved to optimality. A penalty method is employed to deal with the general case. Second, the effect of considering different due date tightness scenarios on the makespan of $F\,|\,no-wait,d_i\,|\,C_{\max}$ is studied by applying the developed PSO to a large number of test problems; the effect of two of the most

widely used dispatching rules on the final solutions is studied and the best dispatching rule is selected to generate initial solutions for the developed PSO algorithm. Finally, although the PSO algorithm is developed to deal with the no-wait flow shop problem with due date constraints, it outperforms competitive methods specifically designed for $F \mid no-wait \mid C_{\max}$. Computational results show that the proposed PSO method is able to find good-quality solutions for the test problems in a reasonable time.

Table 1 – Review of the No-Wait Flow Shop Scheduling Literature

Research	Problem Considered	Proposed Method
Araujoa and Naganoa (2011)	$F \mid no-wait, S_{sd} \mid C_{\max}$	Heuristic
Samarghandi and ElMekkawy (2011)	$F2, S1 \mid no-wait, setup \mid C_{\max}$	Hybrid variable neighborhood search
(Samarghandi and ElMekkawy 2012a)	$F \mid no-wait, setup \mid C_{\max}$	PSO and genetic algorithm
Nagano *et al.* (2012)	$F \mid no-wait, setup \mid \sum C_i$	Evolutionary clustering search
Jolai *et al.* (2012)	No-wait flexible flow shop scheduling problem with sequence dependent setup times	Several metaheuristics
Rabiee *et al.* (2012)	No-wait two-machine flow shop problem with sequence dependent setup times and probable rework	Several metaheuristics
Ying *et al.* (2012)	No-wait flow shop manufacturing cell scheduling problem (FMCSP) with sequence dependent family setup times	Several metaheuristics
Samarghandi and ElMekkawy (2013)	$J2, S1 \mid no-wait, setup \mid C_{\max}$	Genetic algorithm
Nagano *et al.* (2014)	$F \mid no-wait, S_{sd} \mid C_{\max}$	Evolutionary clustering search
Samarghandi and ElMekkawy (2014)	$F \mid no-wait, S_{sd} \mid C_{\max}$	PSO
Samarghandi (Article in Press)	$F, S_Q \mid no-wait, S_{sd} \mid C_{\max}$	Genetic algorithm
Nagano and Araújo (2014)	No-wait flow shop problem with sequence-dependent setup times	Heuristics
Nagano *et al.* (Article in Press)	$F \mid no-wait, S_{sd} \mid \sum C_i$	Heuristics

The rest of the paper is outlined as follows. Section 2 will be devoted to the problem description; also, a mathematical model for the problem will be developed in section 2. Section 3 will explain the proposed PSO algorithm. Furthermore, the employed penalty method will be discussed in section 3. Section

4 will study a number of dispatching rules. Also, computational results are summarized in this section. Section 5 will discuss the concluding remarks and future research directions.

2 Problem Description

2.1 Notation and the Mathematical Model

The following notation is used throughout this paper:

M	Set of machines
$m = \|M\|$	Number of machines
N	Set of jobs
$n = \|N\|$	Number of jobs
J_i	Job i
o_{ij}	j th operation of J_i
p_{ij}	Processing time of the j th operation of J_i on its respective machine
S_i	Starting time of J_i
$S_{o_{ij}}$	Starting time of o_{ij}
d_i	Due date of J_i
π_l	Sequence l
$C_{\max}$	Makespan of π_l

Brackets are used to indicate consecutive jobs, i.e., $S_{[i]}$ refers to the starting time of the job planned to be processed after i th job in a given sequence. Based on the above notations, a mixed integer linear programming model for $F \mid no-wait, d_i \mid C_{\max}$ is as follows:

$$Min\ C_{\max} \tag{1}$$

$$C_{\max} \geq S_{o_{im}} + p_{im};\ \ i = 1,2,...,n \tag{2}$$

$$S_{o_{k1}} + M(1 - x_{ik}) \geq S_{o_{i1}} + p_{i1};\ \ i = 1,2,...,n;\ \ k = 1,2,...,n \tag{3}$$

$$S_{o_{i[j]}} = S_{o_{ij}} + p_{ij};\ \ i = 1,2,...,n;\ \ j = 1,2,...,m-1 \tag{4}$$

$$S_{o_{im}} + p_{im} \leq d_i;\ \ i = 1,2,...,n \tag{5}$$

$$\sum_{i=1}^{n} x_{ij} \leq 1; \quad j = 1,2,...,n \tag{6}$$

$$\sum_{j=1}^{n} x_{ij} \leq 1; \quad i = 1,2,...,n \tag{7}$$

$$x_{ij} + x_{ji} \leq 1; \quad i = 1,2,...,n; \quad j = 1,2,...,n \tag{8}$$

$$\sum_{i=1}^{n} \sum_{j=1}^{n} x_{ij} = n - 1 \tag{9}$$

$$S_{o_{ij}} \geq 0; \quad i = 1,2,...,n; \quad j = 1,2,...,m \tag{10}$$

$$x_{ij} \in \{0,1\}; \quad i = 1,2,...,n; \quad j = 1,2,...,n \tag{11}$$

In this model, the objective function is to minimize the makespan. x_{ij} is a binary variable that is defined as follows:

$$x_{ij} = \begin{cases} 1 \text{ if job } j \text{ is immediately after } i \text{ in the sequence} \\ 0 \qquad\qquad\qquad\qquad\qquad \text{Otherwise} \end{cases} \tag{12}$$

Also, M is a sufficiently large number. (2) defines that makespan equals the finish time of the last operation of the last job. (3) assures that the operations do not overlap; this constraint is binding if J_k is scheduled immediately after J_i in the sequence. (4) imposes the no-wait constraints. (5) is the due date constraint. According to (5), the last operation of each job should finish before its associated due date. Constraints (6), (7), (8) and (9) guarantee that all the jobs will appear exactly once in the sequence. The correctness of this set of constraints can be verified by extracting the values of x_{ij} from any given sequence. Once a sequence of the jobs is available, an algorithm is required to create the actual timetable and compute the makespan of this sequence.

2.2 Timetabling Algorithm

According to the mathematical model of section 2.1, a sequence that violates due date constraints is an infeasible sequence. Consequently, it is assumed that none of the sequences that will be discussed in the rest of this subsection violate the due date constraints.

Algorithm 1: If $c_{\max}$ is the makespan of the partial sequence $\pi = (1,2,...,i)$, and $c'_{\max}$ is the makespan of $\pi' = (1,2,...,i,j)$, then $cont_j = cont_{[i]} = c'_{\max} - c_{\max}$ is called the contribution of job $[i] = j$; and $cont_{[i]}$ can be calculated as follows:

Step 1: Define a counter for the operations of job i and a counter for operations of job $[i]$; call the former counter t and the latter w.

Step 2: Set $t = 2; w = 1$.

Step 3: If $p_{it} \geq p_{[i]w}$, set $t \leftarrow t+1$ and $w \leftarrow w+1$. If $t = m+1$, proceed to step 8; otherwise go back to the beginning of step 3. If $p_{it} < p_{[i]w}$, proceed to step 4.

Step 4: Set $z = \left\{ \min k \,|\, \left(\sum_{l=t}^{k} p_{il} \right) - p_{[i]w} \geq 0 \right\}$ and proceed to step 5. If the value of z cannot be determined, go to step 7.

Step 5: Modify the value of o_{iz} as $p_{iz} = \left(\sum_{l=t}^{z} p_{il} \right) - p_{[i]w}$. Proceed to the next step.

Step 6: Set $w \leftarrow w+1$ and $t \leftarrow z$. If $t = m+1$, go to step 8; otherwise, go back to step 3.

Step 7: Set $cont_{[i]} = \left(\sum_{l=w}^{m} p_{[i]l} \right) - \left(\sum_{l=t}^{m} p_{il} \right)$. Stop.

Step 8: Set $cont_{[i]} = p_{[i]m}$. Stop.

Figure 1 illustrates the above algorithm. Proof of Algorithm 1 can be easily verified using examples and Gantt charts, and therefore is eliminated from this paper. ■

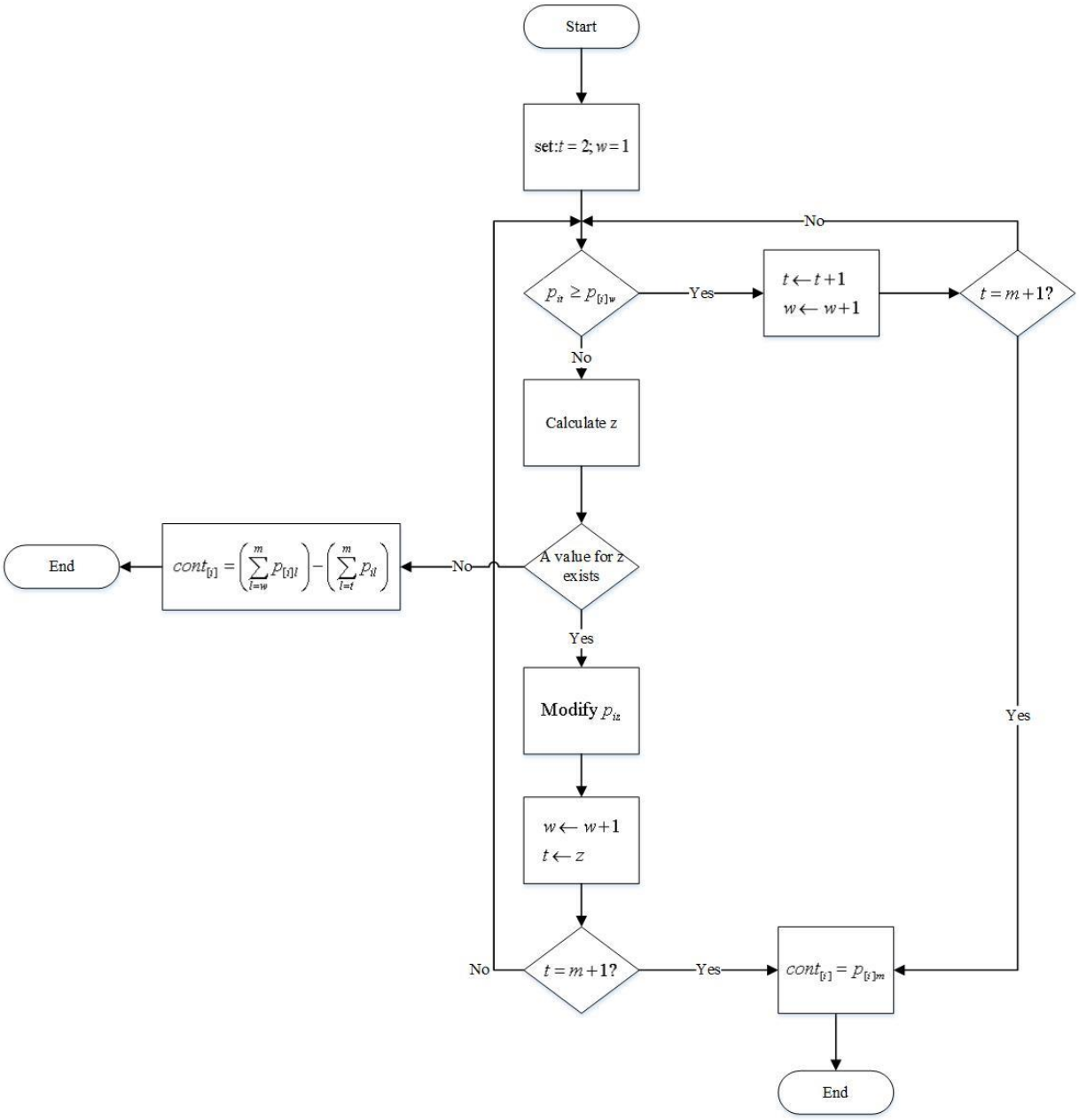

Figure 1 – Illustration of Algorithm 1

Proposition 1: Makespan of permutation $\pi = (1, 2, ..., n)$ can be calculated as

$$c_{\max} = cont_1 + \sum_{i=2}^{n} cont_i = \sum_{j=1}^{m} p_{1j} + \sum_{i=2}^{n} cont_i \, .$$

Proof: Suppose permutation $\pi' = (0,1,2,...,n)$ is such that $p_{0j} = 0; j = 1,...,m$. If algorithm 1 is applied to π' to compute $cont_1$, since $p_{0j} = 0; j = 1,...,m$, the algorithm proceeds to step 4 when it reaches the smallest w such that $p_{1w} > 0$. Moreover, $\left(\sum_{l=w+1}^{m} p_{0l} \right) = 0$, in other words, the index k of step 4 of Algorithm 1 does not exist; therefore, the algorithm proceeds to step 7 and calculates $cont_1$ by (13):

$$cont_{[0]} = cont_1 = \left(\sum_{l=w}^{m} p_{1l} \right) - \left(\sum_{l=w+1}^{m} p_{0l} \right) = \left(\sum_{l=w}^{m} p_{1l} \right) - 0 \xrightarrow{\sum_{j=1}^{w-1} p_{1j} = 0} cont_1 = \left(\sum_{j=1}^{m} p_{1j} \right) \qquad (13)$$

The algorithm then will be applied to jobs $2,3,...,n$ to calculate each job's contribution. ■

Corollary 1: If makespan of $\pi = (1,2,...,i-1,i,i+1,...,j-1,j,j+1,...,n)$ is $c_{\max}$, makespan of $\pi' = (1,2,...,i-1,j,i+1,...,j-1,i,j+1,...,n)$ generated by exchanging jobs i and j is:

$$c'_{\max} = c_{\max} - (cont_i^{\pi} + cont_{i+1}^{\pi} + cont_j^{\pi} + cont_{j+1}^{\pi}) + (cont_i^{\pi'} + cont_{i+1}^{\pi'} + cont_j^{\pi'} + cont_{j+1}^{\pi'}) \qquad (14)$$

Proof: According to Algorithm 1, the contribution of each job is computed based on processing times of the job in comparison with the processing times of its previous job. As a result, when sequence π' is formed by exchanging jobs i and j, the only jobs with a different predecessor compared to sequence π are:

Job	Previous job in π	Previous job in π'
i	$i-1$	$j-1$
$i+1$	i	j
j	$j-1$	$i-1$
$j+1$	j	i

Which proves (14). ■

The proposed algorithm of section 3 employs equation (14) to calculate the objective function of a perturbed sequence once an exchange is applied to a certain sequence. The proposed PSO will be explained in the next section.

3 The Proposed Algorithm

The PSO algorithm has been widely used by researchers to solve combinatorial optimization problems since its introduction in Eberhart and Kennedy (1995) and Kennedy and Eberhart (1997). The PSO algorithm works by systematically moving a number of particles through the search space. At time t, each particle i has a position, $x_i(t)$, and a velocity, $v_i(t)$. A memory stores the current position of the particles as well as their best position. In each step, velocity of the particles is modified according to historical and random information. Velocities, once updated, are used to update the current position of the particles. Then, the PSO evaluates the objective function of the particles at their new position. Since historical data is used in updating the particle velocity, particles tend to return to their historical best position which results in early convergence. To overcome this unwanted phenomenon, different velocity update techniques have been developed. The proposed PSO uses one of the most successful functions available to update the particle velocity.

3.1 Solution Representation

The PSO was originally developed for continuous feasible regions. However, the feasible region of $F|no-wait,d_i|C_{\max}$ is not continuous and consists of the set of permutations of n jobs that do not violate due date constraints. Therefore, the feasible region of $F|no-wait,d_i|C_{\max}$ should be mapped to a continuous region suitable for PSO operations. Numerous coding systems have been developed in the literature that convert discrete feasible regions to a continuous space. This paper uses the SPV coding system that was developed in Tasgetiren *et al.* (2007) and proved to be very efficient for the flow shop problem.

Suppose that particle i at iteration t is given as $X_i^t = [x_{i1}^t, x_{i2}^t, ..., x_{in}^t]$ in which n is the number of jobs in the $F \mid no-wait, d_i \mid C_{\max}$ instance. Although X_i^t is defined in a continuous space, the SPV rule can be applied to this particle to transform it to its corresponding permutation. If $x_{ik}^t < x_{ij}^t; j = 1,2,...,n; j \neq k$, k th location of permutation π will be assigned to job 1. Similarly, if the l th smallest number in X_i^t is seen at location z of X_i^t, job l will be assigned to position z of the permutation π.

Table 2 illustrates the SPV rule when $n = 5$. In this table, x_{i5}^t is the smallest number between x_{ij}^t values. Since the smallest number occurs in position 5, job 1 in permutation π will be assigned to this position. The second smallest number in X_i^t occurs in position 1; therefore job 2 will be placed in this position, and the algorithm continues until all the jobs are assigned to their locations in π. If there are ties, i.e., $x_{ik}^t = x_{ij}^t$, the algorithm randomly selects one of them as the smaller number between the two values to break the tie.

Table 2 – SPV Rule

x_{ij}^t	-0.96	+1.8	+0.43	-0.21	-2.31
Positions	1	2	3	4	5
π	2	5	4	3	1

The above procedure explains the method by which a particle is converted to its corresponding permutation. When it is required to convert a permutation to its corresponding particle, n random numbers will be generated according to (15).

$$x_{ij}^t = x_{\min} + (x_{\max} - x_{\min}) \times r_1 \qquad (15)$$

In (15), $x_{\min} = 0$ and $x_{\max} = 4$ and $r_1 \sim U(0,1)$. Afterward, the smallest generated number will be assigned to the position of job 1. The second smallest generated number will be assigned to the position of job 2, and so on. Once again, ties will be dealt with using a random procedure.

3.2 Generating Initial Solutions

The proposed PSO requires a number of initial solutions to begin its exploration in the feasible region. Once the initial solutions are generated and their makespans are calculated as described in section 2.2, the method of section 3.1 will be employed to convert the permutations into particles. No particle is born or destroyed during the search. Note that based on the one-to-one SPV mapping described in section 3.1, a particle is identical to a complete permutation. Therefore, these two words can be used interchangeably during the rest of the paper. The number of initial solutions, represented by I hereafter, is a parameter of the algorithm and will be set by the user. Once initial solutions are generated, they will be converted to particles ($X_i^0; i = 1,2,...,I$) and the PSO algorithm starts the search.

It should be noted that different methods exist for generating the initial solutions. Several papers in the literature have studied the performance of the priority dispatching rules under different assumptions and with different performance measures. In section 4.2, two methods of generating initial solutions and their effects on the results of the proposed algorithm will be studied. These methods include random permutations and permutations created according to the earliest due date dispatching rule.

3.3 Infeasible Solutions and Penalties

As the developed mathematical model of section 2.1 indicates, due date requirements are considered as hard constraints in the model. In other words, schedules that violate these constraints will be marked as infeasible by the model. However, the proposed PSO allows the violation of the due date constraints with the hope that the violations will be removed during the search if a penalty is imposed to the objective function. Consequently, the proposed PSO modifies the objective function of the mathematical model to (16) and removes the due date constraints from the model.

$$Min\ C_{\max} + \lambda \sum_{i=1}^{n} U_i \times \left(S_{o_{im}} + p_{im} - d_i \right) \tag{16}$$

In which:

$$U_i = \begin{cases} 1 & \text{if } S_{o_{im}} - p_{im} > d_i \\ 0 & \text{Otherwise} \end{cases} \tag{17}$$

Section 4.1 introduces a design of experiments method based on Taguchi approach to optimize the value of λ and its effect on the makespan of the final solutions of the proposed PSO.

3.4 The Proposed PSO Algorithm

Once the initial particles are generated, the algorithm requires an initial velocity vector for each particle to update the position of the particles and continue the search in the feasible region of the problem. Initial velocities are generated by (18).

$$v_{ij}^0 = v_{\min} + (v_{\max} - v_{\min}) \times r_2 \tag{18}$$

In (18), $v_{\min} = -4$, $v_{\max} = +4$, and $r_2 \sim U(0,1)$. Values of $v_{ij}^t; j - 1,2,...,n$ are bounded by (19) during all of the iterations of the algorithm.

$$v_{ij}^t \in [-4, +4] \tag{19}$$

If $v_{ij}^t > 4$, then the algorithm modifies this velocity to +4; and if $v_{ij}^t < -4$, then the velocity will be modified to -4. Once $V_i^0; i = 1,2,...,I$ are generated, the position of the particles is updated using (20).

$$\begin{aligned} &X_i^1 = [x_{i1}^0 + v_{i1}^0, x_{i2}^0 + v_{i2}^0, ..., x_{in}^0 + v_{in}^0] \\ &i = 1,2,...,I \end{aligned} \tag{20}$$

In general, if $V_i^t = [v_{i1}^t, v_{i2}^t, ..., v_{in}^t]$ is the velocity vector that accompanies X_i^t, then X_i^{t+1}, the position of the i th particle in iteration $t+1$, is found by equation (21).

$$X_i^{t+1} = [x_{i1}^t + v_{i1}^t, x_{i2}^t + v_{i2}^t, ..., x_{in}^t + v_{in}^t] \tag{21}$$

The algorithm stores the personal best or the best position of particle $i; i = 1,2,...,I$ during the search in $P_i; i = 1,2,...,I$, and the global best or the best position of all the particles during the search in G. The algorithm evaluates the objective function of all the particles in each iteration and updates the values of P_i and G if required. The equation that updates the velocity vectors in each iteration is as follows:

$$V_i^{t+1} \leftarrow \chi\left(wV_i^t + cr(P_i - X_i^t)\right) \tag{22}$$

In which w and χ, inertia weight and constriction coefficient, are calculated as follows:

$$\begin{aligned} w &= w_{\max} - \frac{w_{\max} - w_{\min}}{Iter} \times t \\ \chi &= \frac{2}{c - 2 + \sqrt{c^2 - 4c}}; c > 4 \end{aligned} \tag{23}$$

$w_{\max}$ and $w_{\min}$ are two parameters set by the user, $Iter$ is the total number of iterations, and t is the number of the current iteration. If objective function value of X_i^{t+1} is greater than the objective function value of X_i^t for $i = 1,2,...,I$, the proposed PSO algorithm applies the local search sub-algorithm to X_i^{t+1}. This sub-algorithm is described in the next section. The PSO algorithm stops when $t = Iter$, and returns G as the final solution.

3.5 Local Search

The local search sub-procedure first converts the particle in its corresponding permutation π^{t+1}, once the particle X_i^{t+1} is sent to this sub-algorithm, and computes the objective function value of this permutation. Suppose that $\pi^{t+1} = (1,2,...,n)$; the following steps describe the local search algorithm:

1. Select i such that $cont_i > cont_j; j = 1,2,...,n; j \neq i$; if more than one job can be selected with this condition, randomly select one of them. Then define the lateness of each job as $L_j = d_j - S_{o_{jm}} - p_{jm}; j = 1,2,...,n$.

2. If $L_i > 0$, set $l = 1$ and proceed to step 3. Otherwise, go to step 5.
3. In sequence π^{t+1}, form $A = \{ j \mid j \text{ is on the right of } i, L_j \le 0 \}$. Select job $k \in A$ with the greatest contribution. If $A = \phi$, select the job that has the greatest contribution after J_i. Suppose that job k is selected. If $l <= R$ go to step 4.
4. Exchange the places of jobs i and k in the permutation. If the exchange results in a reduction in the value of the objective function, accept this exchange and proceed to step 7; otherwise, set $l \leftarrow l + 1$, reverse the exchange, remove job k from the comparisons, and return to step 3.
5. In sequence π^{t+1}, form $A = \{ j \mid j \text{ is on the left of } i, L_j > 0 \}$. Select job $k \in A$ with the greatest contribution. If $A = \phi$, select the job that has the greatest contribution after J_i. Suppose that job k is selected. If $l <= R$ go to step 6.
6. Exchange the places of jobs i and k in the permutation. If the exchange results in a reduction in the value of the objective function, accept this exchange and proceed to step 7; otherwise, set $l \leftarrow l + 1$, reverse the exchange, remove job k from the comparisons, and return to step 5.
7. Use the SPV ruling of section 3.1 to code the resulted permutation. Submit this code to the PSO algorithm.

R is a parameter of the algorithm and will be determined by the user. As mentioned earlier, this algorithm uses the results of Corollary 1 in order to evaluate the objective function of the new permutation once an exchange is executed. Performance of the proposed PSO will be examined in the next section.

4 Computational Results

Microsoft Visual C++ 2008 was chosen to code the PSO algorithm. All the test problem instances are solved on a PC equipped with a 3GHz Intel Pentium IV CPU and 2 GB of RAM. To perform the

computational analysis, a number of test problems were selected from the literature available for $F|no-wait|C_{\max}$. Car problems were introduced by Carlier (1978) and Rec problems were generated by Reeves (1995). These test problems are available from the OR-Library[1]. From the set of selected problems, Car problems have optimal solutions, while Rec problems do not have optimal solutions. To generate the due dates, a formula similar to Tari and Olfat (2014) was adopted:

$$d_i \sim U\left(\sum_{j=1}^{m} p_{ij}, (6-TF)\times\sum_{j=1}^{m} p_{ij}\right) \tag{24}$$

In which TF is the tightness factor of the due dates. For each test problem, 4 different tightness factor settings were considered according to (25) which results in a total of 60 test problems for $F|no-wait,d_i|C_{\max}$; eight Car problems and seven Rec problems with four different tightness factors for each of these problems. Once the due dates are generated, the resulting problems are called Car+DD and Rec+DD.

$$\begin{aligned} TF_1 &= 1 \\ TF_2 &= 2 \\ TF_3 &= 3 \\ TF_4 &= 4 \end{aligned} \tag{25}$$

4.1 Tuning the Algorithm Parameters

As seen in section 3, the developed PSO has 7 control parameters; λ, the penalty coefficient in (16) is also considered a parameter of the algorithm. These parameters must be tuned to obtain the best performance of the developed algorithms. A Taguchi-based design of experience approach has been followed to determine the effect of the different values of the parameters on the performance of the algorithm. Accordingly, 3 different test problems with 4 different tightness factors were chosen; the considered test problems were Car01+DD, Rec13+DD and Rec31+DD. In addition 3 different values for each parameter were selected. This leads to a Taguchi design with 7 factors and 3 levels for each factor.

[1] Beasley, J.E. *OR-Library: distributing test problems by electronic mail*. July 2009 [cited 2014 March]; Available from: http://people.brunel.ac.uk/~mastjjb/jeb/info.html.

Considered values for the parameters can be found in Table 3. The parameter values were selected in a trade-off between the required solution time of the algorithm and the capability of the algorithm for diversification of the solutions and intensification around the promising results.

Table 3 – Parameter Combinations

Parameter	Value 1	Value 2	Value 3
I	$\frac{n}{2}$	n	$2n$
$w_{\min}$	0.4	0.5	0.6
$w_{\max}$	0.9	1.0	1.1
c	4.25	4.5	4.75
R	40	60	80
$Iter$	$10n$	$15n$	$20n$
λ	1	3	5

The interaction between the parameters were assumed to be negligible. Accordingly, the Taguchi design requires 27 different combinations of the levels of the considered factors to generate reliable results. To supply the Taguchi analysis with replications and improve the robustness of the design, each test problem and tightness factor combination was solved 5 times by the PSO algorithm. According to (26), this leads to 1620 independent runs of the algorithm.

$$1620 = 3\,\text{Problems} \times 4\,\text{Tightness Factor} \times 5\,\text{Replications} \times 27\,\text{Factor/Level Combinations} \quad (26)$$

Once the objective function values of the mentioned 1620 runs of the algorithm were obtained, (27) was used to calculate the relative deviation (RD) of each objective function value from the best objective function value in each considered combination of the test problem and due date tightness factor.

$$RD^{\text{Prob}} = \left| \frac{OFV^{\text{Prob}} - OFV^{\text{Best}}}{OFV^{\text{Best}}} \right| \quad (27)$$

In which RD^{Prob} is the relative deviation of the considered test problem, OFV^{Prob} is the objective function value of the considered test problem, and OFV^{Best} is the minimum objective function value among the 5 replications of the considered test problem. Afterward, the results were analyzed with the Taguchi design of experiments approach. Figure 2 presents the main effects plot for the means, and Figure 3 depicts the main effects plot for the signal-to-noise ratios. Accordingly, it can be noticed that ties exist in selecting the best combination for the parameter values (minimum values for signal-to-noise and mean do not occur at the same parameter values). However, analysis of variance shows that the parameters I and $W_{\max}$ are not statistically significant. Therefore, it is possible to predict the response variable (RD^{Prob}) by the remaining parameters and using the Taguchi method. Once the prediction is performed, extra runs of the algorithm verify that the following combination for the parameters is more desirable; this is due to the fact that according to (27), smaller values for RD^{Prob} are more desirable. Interested reader is referred to Montgomery (2008) for more information about Taguchi method.

$$
\begin{aligned}
&I = \frac{n}{2} \\
&W_{\min} = 0.4 \\
&W_{\max} = 1.1 \\
&c = 4.5 \\
&R = 40 \\
&Iter = 20n \\
&\lambda = 3
\end{aligned}
\tag{28}
$$

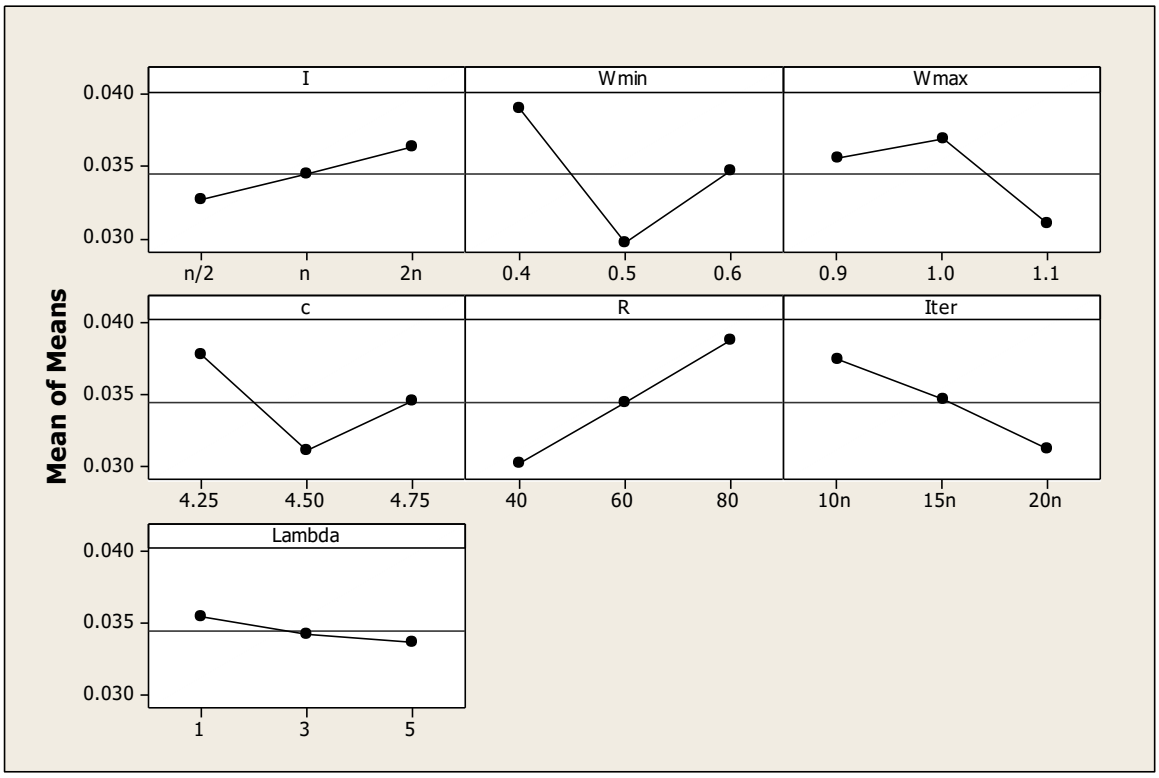

Figure 2 – Main Effects Plot for the Means

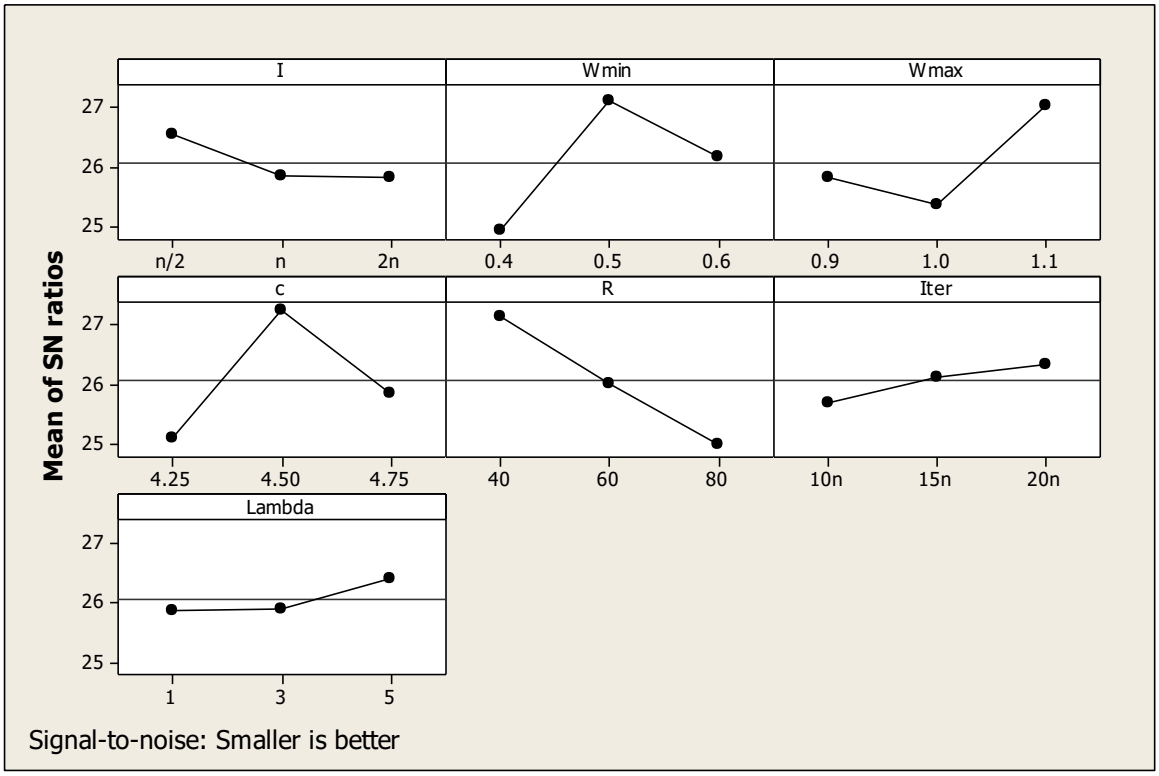

Figure 3 – Main Effects Plot for the Signal-to-Noise Ratio

4.2 Dispatching Rules

As mentioned in section 1, Hunsucker and Shah (1992), Brah (1996) and Tari and Olfat (2014) study the efficiency of a number of dispatching rules for flow shop problem with different settings in the presence of due date constraints. Hunsucker and Shah (1992) show that when scheduling a constrained flow shop with multiple processors and due dates, the first-in-first-out priority rule is superior when the objective function is to minimize the mean tardiness. However, when minimizing the number of tardy jobs, a superior priority rule cannot be established. Tari and Olfat (2014) conclude that for tardiness flow shop models with intermediate due dates, simpler priority rules such as shortest processing time usually lead to better solutions; however, such cannot be concluded for the case of traditional tardiness flow shop with intermediate due dates.

Two different priority dispatching rules were considered in this paper, namely, Earliest Due Date (EDD) and Random Dispatching (RD). Five test problems were selected from the set of the considered test problems of this paper, and were solved using the proposed algorithm ($\lambda = 3$). Initial solutions of these problems were generated based on the EDD and RD rules. Table 4 summarizes the objective function values of the initial solutions of these test problems under EDD and RD. Figure 4 illustrates the progress of the proposed PSO when different dispatching rules are applied to Rec01+DD.

Table 4 and Figure 4 demonstrate that the quality of the initial solutions as well as the quality of the final answer of the EDD rule were both significantly superior to the RD rule. In other words, the algorithm generally produces dominant final solutions when the search commences from a more promising area of the feasible region. Therefore, in sections 4.3 and 4.4, EDD was selected as the superior method to generate the initial solutions of the proposed PSO and to initialize the search.

Table 4 – Quality of Solutions with Different Dispatching Rules

		EDD			RD			EDD			RD		
Problem	**TF**	**Initial OFV**	**Makespan**	**Penalty**	**Initial OFV**	**Makespan**	**Penalty**	**Final OFV**	**Makespan**	**Penalty**	**Final OFV**	**Makespan**	**Penalty**
Car01 +DD	**1**	**9,487**	9,487	0	29,653	12,106	17,547	**8,298**	8,298	0	8,343	8,343	0
	2	**10,430**	10,430	0	29,057	11,186	17,871	**8,168**	8,168	0	8,338	8,338	0
	3	**26,912**	10,757	16,155	50,500	9,724	40,776	**11,356**	9,316	2,040	12,826	9,526	3,300
	4	**75,332**	10,499	64,833	125,714	12,542	113,172	**48,940**	8,830	40,110	51,658	9,172	42,486
Car05 +DD	**1**	**11,547**	11,547	0	14,296	14,296	0	**9,159**	9,159	0	9,188	9,188	0
	2	**11,078**	11,078	0	24,873	11,793	13,080	**9,454**	9,454	0	**9,454**	9,454	0
	3	**12,997**	11,950	1,047	58,714	12,364	46,350	**11,443**	11,347	96	11,517	11,127	390
	4	**52,021**	11,179	40,842	135,265	13,063	122,202	44,182	10,612	33,570	**39,223**	10,282	28,941
Rec01 +DD	**1**	**1,837**	1,837	0	10,582	2,152	8,430	**1,672**	1,672	0	1,706	1,706	0
	2	**7,446**	1,954	5,492	24,322	2,083	22,239	**3,119**	1,718	1,401	3,430	1,732	1,698
	3	**20,574**	1,950	18,624	29,856	2,118	27,738	15,627	1,683	13,944	**14,452**	1,672	12,780
	4	**33,987**	2,064	31,923	39,358	2,038	37,320	**23,704**	1,603	22,101	23,978	1,646	22,332
Rec19 +DD	**1**	**4,448**	3,866	582	27,989	4,106	23,883	3,224	3,224	0	**3,193**	3,193	0
	2	**24,427**	3,811	20,616	83,731	4,396	79,335	**5,449**	3,229	2,220	6,307	3,334	2,973
	3	**67,840**	3,883	63,957	83,353	4,228	79,125	**40,139**	3,155	36,984	43,124	3,206	39,918
	4	**119,819**	4,286	115,533	140,435	4,433	136,002	**66,052**	3,211	62,841	68,314	3,031	65,283
Rec37 +DD	**1**	**449,348**	11,987	437,361	609,434	12,662	596,772	173,978	8,996	164,982	**155,496**	9,012	146,484
	2	**601,202**	11,789	589,413	834,140	13,289	820,851	**297,457**	8,917	288,540	312,406	8,944	303,462
	3	952,484	12,860	939,624	**868,507**	12,148	856,359	**477,639**	8,907	468,732	480,874	8,881	471,993
	4	**1,067,133**	12,339	1,054,794	1,150,176	13,047	1,137,129	**621,869**	9,065	612,804	631,522	8,818	622,704

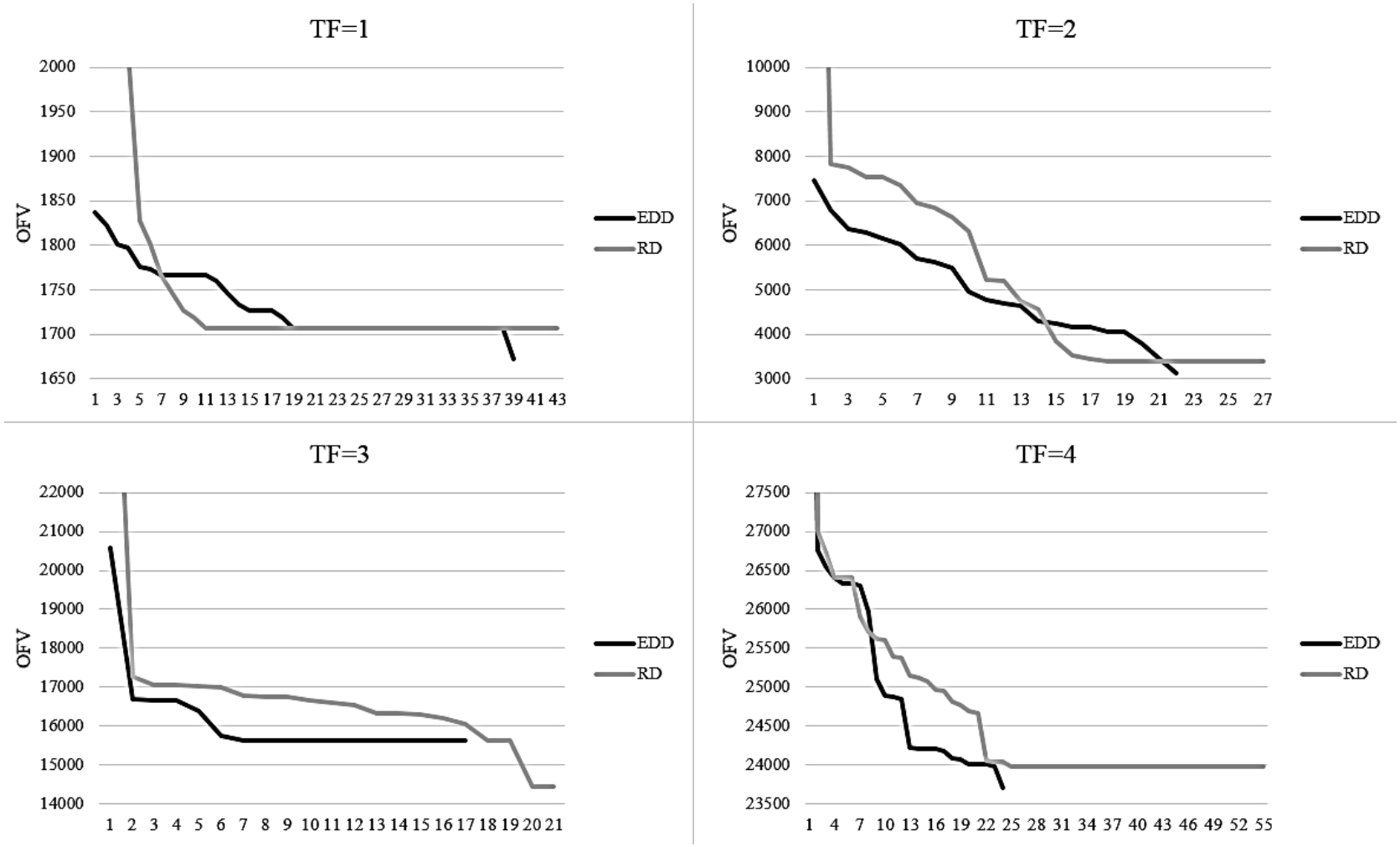

Figure 4 – Progress of the Developed PSO When Different Dispatching Rules Applied to Rec01+DD

Table 5 – Computational Results of the Problems with Optimal Solution

Prob.	Size m*n	No-Wait OFV[1,2]	TF[3]	Optimum Solution[4]	Best OFV	Best C_{max}	Average OFV	Average C_{max}
Car1 +DD	11*5	8,142	1	8,152	8,152	8,152	8,178	8,178
			2	8,164	8,164	8,164	8,194	8,194
			3	NFS[5]	11,356	9,316	12,479	9,476
			4	**NFS**	48,986	8,948	49,527	8,843
Car2 +DD	13*4	8,242	1	8,646	8,471	8,471	8,523	8,523
			2	9,921	9,002	9,002	9,213	9,213
			3	NFS	17,066	9,449	18,699	9,482
			4	**NFS**	39,145	9,037	40,370	9,110
Car3 +DD	12*5	8,866	1	9,264	9,084	9,084	9,084	9,084
			2	9,120	9,220	9,220	9,319	9,319
			3	NFS	11,409	9,696	12,460	9,995
			4	**NFS**	67,267	9,667	69,188	9,876
Car4 +DD	14*4	9,195	1	10,305	9,746	9,746	10,228	10,167
			2	NFS	16,286	10,979	16,310	10,992
			3	NFS	23,136	11,241	24,500	11,025
			4	**NFS**	74,245	10,630	76,132	10,359
Car5 +DD	10*6	9,159	1	9,159	9,159	9,159	9,196	9,196
			2	9,454	9,558	9,558	9,695	9,695
			3	**11,537**	11,537	11,537	11,537	11,537
			4	**NFS**	39,223	10,282	40,181	10,305
Car6 +DD	8*9	9,690	1	**9,690**	9,690	9,690	9,758	9,758
			2	**9,690**	9,690	9,690	9,690	9,690
			3	**9,690**	9,690	9,690	9,837	9,837
			4	**NFS**	11,429	11,090	11,429	11,090
Car7 +DD	7*7	7,705	1	**7,705**	7,705	7,705	7,803	7,803
			2	**7,705**	7,705	7,705	7,705	7,705
			3	**7,705**	7,705	7,705	7,705	7,705
			4	**NFS**	18,014	8,816	18,014	8,816
Car8 +DD	8*8	9,372	1	**9,372**	9,372	9,372	9,387	9,387
			2	**9,372**	9,372	9,372	9,372	9,372
			3	**9,573**	9,573	9,573	9,573	9,573
			4	**NFS**	14,213	11,552	14,213	11,552

[1] OFV: Objective Function Value

[2] All the OFVs in this column belong to the optimum solution of $F\,|\,no-wait\,|\,C_{max}$

[3] TF: Tightness Factor

[4] Optimum solution of $F\,|\,no-wait, d_i\,|\,C_{max}$; bold numbers are proven optimum solutions.

[5] No Feasible Solutions Found

4.3 Problems with Optimal Solutions

Table 5 presents the computational results of Car01+DD through Car08+DD. These problems generally have fewer jobs compared to the set of Rec+DD problems. As a result, it is possible to solve many of them to optimality by means of the mathematical model of section 2.1. These problems were solved

using IBM ILOG CPLEX. It should be noted that the solver was not able to find the optimal solution of some of the test problems in less than 2 hours. Therefore, the reported solutions of the model are the best solutions that were obtained after 2 hours of execution. As a result, the solutions of the proposed PSO may be better than the solutions of the mathematical model. The proven optimum solutions appear in boldface. One can verify that the proposed algorithm is very competitive and is able to produce the optimal solutions in most cases. Average CPU time to solve the mathematical model was 3,354 seconds, and the algorithm's average CPU time was 2.4 seconds.

4.4 Problems without Optimal Solutions

The set of 21 Rec problems were solved with the developed algorithm and without considering the due date constraints; the results will be compared to the competitive methods in the literature developed for $F|no-wait|C_{\max}$. Table 6 summarize the computational results of the proposed PSO for the problems without optimal solutions in $F|no-wait|C_{\max}$ environment. In this table the first two columns present the test problem and the size of the problem. The third column belongs to the makespans of Rajendran (1994), which have been traditionally used for comparison purposes. Next two columns present the objective function of the proposed algorithm and the relative deviation between the reported objective function and the makespan of Rajendran (1994). Each test problem was solved five times but only the best solution is reported. This approach keeps the results comparable to the competitive methods from the literature. The reported makespans were obtained by assigning $\lambda = 0$ when solving the test problems, which is analogous to removing the penalty term of equation (16). The next two columns belong to the solution found for the test problem in $F|no-wait|C_{\max}$ environment by the TS+PSO of Samarghandi and ElMekkawy (2012b) (a hybrid of the tabu search and PSO). Next columns present the relative deviation between the proposed solutions of the competitive methods from the literature and the makespan of Rajendran (1994). It can be verified that although the proposed algorithm is specifically designed to deal

with the due date constraints, its computational results are very competitive. Relative deviation is calculated by:

$$\text{Relative Deviation} = \max_{i=1,\ldots,5}\left\{100\times\left|\frac{c^{i}_{\max}-c^{*}_{\max}}{c^{*}_{\max}}\right|\right\} \tag{29}$$

In which $c^{i}_{\max}$ is the proposed makespan of the algorithm and $c^{*}_{\max}$ is the proposed makespan of Rajendran (1994).

To verify the effectiveness of the proposed PSO for the problems with due date constraints, seven test problems from the set of Rec problems with different sizes were selected. According to (24), four different tightness factors were employed to generate four random due dates for each test problem. Table 7 summarizes the computational results of the proposed PSO for the problems without optimal solutions when due date constraints are added to the test problems. This table also presents the proposed solution of Samarghandi and ElMekkawy (2012b) for Rec+DD test problems when TS+PSO of Samarghandi and ElMekkawy (2012b) is modified to accommodate due date constraints. The reason for selecting TS+PSO of Samarghandi and ElMekkawy (2012b) for comparison with the proposed algorithm of this paper is that according to Table 6 TS+PSO generates the best solutions for $F\,|\,no-wait\,|\,C_{\max}$ for most of the considered test problems. Each test problem is solved five times. For the proposed algorithm the best, average and worst objective functions and makespans as well as the CPU times are reported; for the TS+PSO, only the best solutions are reported. The proposed PSO takes about 20 seconds of CPU time to generate solutions for problems with 75 jobs and 20 machines; average CPU time for the same problem when solved with TS+PSO was 48 seconds. Deviation between the best objective function value of the proposed PSO and the best objective function value of the TS+PSO is calculated as follows:

$$\text{Deviation} = \frac{OFV^{PSO}_{Best}-OFV^{TS+PSO}_{Best}}{OFV^{PSO}_{Best}}\times 100 \tag{30}$$

Accordingly, negative values mean that the objective function of the proposed PSO is superior to the objective function of the TS+PSO, and vice versa. The average deviation indicates that the proposed PSO generates superior objective functions compared to TS+PSO. It is expected that as the value of λ increases, total lateness values decrease and makespans slightly increase. This is a natural outcome of imposing the objective function with more penalty when lateness occurs; when penalties associated with the lateness increase, the algorithm will sacrifice the makespan in order to decrease the lateness and improve the objective function value.

Table 6 - Comparison of the Proposed Algorithm with Competitors for $F \mid no-wait \mid C_{\max}$

		Rajendran (1994)	Proposed Algorithm		Samarghandi and ElMekkawy (2012b)		Liu *et al.* (2007)	Schuster and Framinan (2003)		Grabowski and Pempera (2005)				
Prob.	**Size m*n**	**OFV**	**OFV**	**Relative Deviation[1]**	**OFV**	**Relative Deviation**	**HPSO**	**VNS**	**GASA**	**DS**	**DS+M**	**TS**	**TS+M**	**TS+MP**
Rec01	20,5	1,590	1,528	3.90	1,528	3.90	3.77	2.77	3.96	3.71	3.58	**4.03**	3.96	3.96
Rec03	20,5	1,457	**1,361**	7.05	**1,361**	7.05	6.59	4.32	4.46	3.43	4.43	6.59	6.59	6.59
Rec05	20,5	1,637	**1,511**	8.34	**1,511**	8.34	7.39	7.03	6.90	5.62	5.62	7.39	7.64	7.70
Rec07	20,10	2,119	2,043	3.72	2,042	3.77	3.63	2.31	3.45	1.09	1.08	3.63	3.63	3.63
Rec09	20,10	2,141	2,043	4.80	**2,027**	5.62	4.58	2.38	4.48	3.60	3.60	4.62	4.58	4.58
Rec11	20,10	1,946	1,888	3.07	**1,881**	3.46	3.34	1.54	3.34	1.44	1.44	3.34	3.34	3.34
Rec13	20,15	2,709	**2,545**	6.44	**2,545**	6.44	6.05	5.76	5.65	3.43	4.43	6.05	6.05	6.05
Rec15	20,15	2,691	**2,529**	6.41	**2,529**	6.41	6.02	5.91	6.02	4.83	4.83	5.91	6.02	5.91
Rec17	20,15	2,740	**2,587**	5.91	**2,587**	5.91	5.58	5.15	5.47	5.51	5.51	5.58	5.58	5.58
Rec19	30,10	3,157	2,864	10.23	**2,861**	10.35	9.15	7.57	5.45	7.70	7.44	9.72	9.25	9.38
Rec21	30,10	3,015	2,843	6.05	**2,822**	6.84	5.70	4.21	2.22	3.68	4.68	6.37	6.30	6.17
Rec23	30,10	3,030	2,707	11.93	**2,700**	12.22	10.80	10.70	6.70	7.29	7.29	10.76	10.73	10.89
Rec25	30,15	3,835	3,596	6.65	**3,593**	6.74	5.71	5.45	2.69	3.08	3.08	5.97	6.31	6.21
Rec27	30,15	3,655	3,434	6.44	**3,431**	6.53	6.13	5.83	2.60	3.64	3.64	5.64	6.10	5.83
Rec29	30,15	3,583	**3,291**	8.87	**3,291**	8.87	7.81	7.23	3.99	7.23	7.36	7.94	8.28	7.94
Rec31	50,10	4,631	**4,336**	6.80	**4,336**	6.80	5.92	4.71	-2.72	3.76	3.78	5.90	6.13	6.22
Rec33	50,10	4,770	4,496	6.09	**4,466**	6.81	5.51	5.35	-4.78	1.97	2.01	5.51	6.31	6.37
Rec35	50,10	4,718	4,441	6.24	**4,417**	6.81	6.02	5.51	-3.67	4.94	4.94	6.08	6.17	5.91
Rec37	75,20	8,979	8,170	9.90	**8,081**	11.11	8.89	10.00	-5.89	7.80	7.92	9.41	9.49	9.36
Rec39	75,20	9,158	8,593	6.58	**8,517**	7.53	6.79	5.32	-8.80	4.97	5.12	7.00	6.99	6.91
Rec41	75,20	9,344	8,627	8.31	8,520	9.67	7.94	7.41	-6.79	6.08	6.08	8.78	8.57	8.82
Average		**N/A**	**N/A**	**6.84**	**N/A**	**7.36**	**6.35**	**5.55**	**1.65**	**4.51**	**4.66**	**6.49**	**6.57**	**6.54**

[1] Larger numbers are more desirable

Table 7 – Computational Results of the Large-Instance Problems

			The Proposed Algorithm										TS+PSO Samarghandi and ElMekkawy (2012b)			Deviation Between OFV of the Proposed PSO and TS+PSO
Prob.	**Size m*n**	**TF**	**Best OFV**	**Best Makespan**	**Total Lateness**	**Worst OFV**	**Worst Makespan**	**Total Lateness**	**Average OFV**	**Average Makespan**	**Average Lateness**	**Average CPU Time**	**Best OFV**	**Best Makespan**	**Total Lateness**	
Rec 01+ DD	20*5	1	**1,668**	1,668	0	1,697	1,697	0	1,684	1,684	0	3.03	1,682	1,682	0	-0.84
		2	**3,119**	1,718	467	4,685	1,763	974	3,782	1,737	681	3.59	3,442	1,747	1,695	-10.36
		3	**14,809**	1,597	4,404	15,683	1,682	4,667	15,237	1,665	4,524	3.37	15,132	1,671	13,461	-2.18
		4	**23,688**	1,605	7,361	24,214	1,624	7,530	23,994	1,646	7,449	3.18	23,806	1,612	22,194	-0.50
Rec 07+ DD	20*10	1	**2,127**	2,127	0	2,141	2,141	0	2,134	2,134	0	4.27	2,142	2,142	0	-0.71
		2	**2,138**	2,138	0	2,228	2,228	0	2,191	2,191	0	4.46	2,241	2,241	0	-4.82
		3	2,869	2,299	190	2,869	2,299	190	2,869	2,299	190	4.10	**2,461**	2,260	201	14.22
		4	18,308	2,171	5,379	19,379	2,174	5,735	18,898	2,199	5,566	4.78	**17,725**	2,248	15,477	3.18
Rec 13+ DD	20*15	1	**2,553**	2,553	0	2,670	2,670	0	2,595	2,595	0	5.19	2,711	2,711	0	-6.19
		2	2,651	2,651	0	2,655	2,655	0	2,652	2,652	0	5.71	**2,607**	2,607	0	1.66
		3	**2,648**	2,648	0	2,773	2,773	0	2,711	2,711	0	6.78	2,681	2,681	0	-1.25
		4	13,456	2,785	3,557	15,079	2,764	4,105	14,417	2,787	3,876	5.80	**12,154**	2,743	9,411	9.68
Rec 19+ DD	30*10	1	**3,087**	3,087	0	3,165	3,165	0	3,117	3,117	0	8.25	3,203	3,203	0	-3.76
		2	**4,799**	3,233	522	5,436	3,252	728	5,143	3,220	641	8.58	5,046	3,228	1,818	-5.15
		3	**35,067**	3,075	10,664	36,905	3,119	11,262	36,037	3,101	10,979	6.86	35,529	3,075	32,454	-1.32
		4	**69,262**	3,178	22,028	72,850	3,166	23,228	71,156	3,160	22,665	5.82	70,971	3,147	67,824	-2.47
Rec 25+ DD	30*15	1	**3,710**	3,710	0	3,733	3,733	0	3,722	3,722	0	8.39	3,747	3,747	0	-1.00
		2	**3,878**	3,878	0	3,986	3,983	1	3,926	3,925	0	10.23	4,049	4,049	0	-4.41
		3	**14,927**	3,917	3,670	17,740	4,027	4,571	16,418	3,940	4,159	7.60	17,498	3,965	13,533	-17.22
		4	**63,080**	3,893	19,729	66,136	3,946	20,730	64,413	3,922	20,163	6.75	65,551	3,934	61,617	-3.92
Rec 31+ DD	50*10	1	**46,174**	4,678	13,832	52,501	4,837	15,888	48,348	4,747	14,534	10.45	53,696	4,868	48,828	-16.29
		2	97,712	4,676	31,012	101,946	4,809	32,379	99,773	4,708	31,688	12.36	**97,466**	4,637	92,829	0.25
		3	143,265	4,629	46,212	149,169	4,734	48,145	146,695	4,696	47,333	8.96	**140,540**	4,697	135,843	1.90
		4	215,235	4,677	70,186	218,231	4,778	71,151	216,660	4,710	70,650	9.81	**209,627**	4,589	205,038	2.61
Rec 37+ DD	75*20	1	**155,357**	8,786	48,857	160,774	8,806	50,656	158,361	8,797	49,855	25.28	160,274	8,804	151,470	-3.16
		2	**292,245**	8,787	94,486	298,350	8,592	96,586	295,868	8,725	95,714	25.40	314,104	8,842	305,262	-7.48
		3	**459,561**	8,904	150,219	464,090	8,813	151,759	462,341	8,866	151,158	24.14	478,000	8,986	469,014	-4.01
		4	**616,521**	8,676	202,615	636,607	8,797	209,270	625,643	8,743	205,633	23.72	631,471	8,791	622,680	-2.42
Average			**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**9.17**	**N/A**	**N/A**	**N/A**	**-2.36**

5 Conclusions

This paper considered the scheduling problem of $F|no-wait,d_i|C_{\max}$. The problem is strongly NP-Hard. A mathematical model of the problem was developed, and the problem was reduced to a permutation problem. An efficient algorithm was developed to generate timetables for $F|no-wait|C_{\max}$ when a permutation of jobs is given. A particle swarm algorithm was developed to deal with the general cases of the $F|no-wait,d_i|C_{\max}$ problem. A new local search approach was introduced to further improve the computational results of the proposed PSO. A design of experiments approach using Taguchi method was employed to tune the parameters of the developed algorithm. Two dispatching rules were investigated; accordingly, the earliest due date dispatching rule was selected to generate the initial solutions necessary for initializing the proposed PSO.

A thorough computational analysis was performed on the small- and large-instance test problems available in the literature. Computational analysis consisted of different penalty coefficients and due date tightness factors. Optimal solution of several small-instance test problems were found by means of the developed mathematical model. The developed PSO proved to be very efficient for problems with and without optimal solutions. The algorithm was able to generate good-quality solutions for the test problems in a reasonable time.

A possibility for future research is finding lower and upper bounds for $F|no-wait,d_i|C_{\max}$. In contrast, finding feasible solutions for problems with tight due dates can be a challenge. Therefore, developing an approach that is able to efficiently generate feasible solutions is very promising. Moreover, a thorough study of the different dispatching rules and their effects on the quality of the final solutions would be useful.

6 Acknowledgements

The authors would like to thank the anonymous referees for their precious comments that helped with improving the quality of this paper.

7 References

Araujoa, D. C. and M. S. Naganoa (2011). "A new effective heuristic method for the no-wait flowshop with sequence-dependent setup times problem." International Journal of Industrial Engineering Computations **2**: 155 - 166.

Błażewicz, J., E. Pesch, M. Sterna and F. Werner (2005). "The two-machine flow-shop problem with weighted late work criterion and common due date." European Journal of Operational Research **165**(2): 408-415.

Błażewicz, J., E. Pesch, M. Sterna and F. Werner (2008). "Metaheuristic approaches for the two-machine flow-shop problem with weighted late work criterion and common due date." Computers & Operations Research **35**(2): 574-599.

Brah, S. (1996). "A comparative analysis of due date based job sequencing rules in a flow shop with multiple processors." Production Planning & Control **7**(4): 362-373.

Carlier, J. (1978). "Ordonnancements a contraintes disjonctives." RAIRO Recherche Operationnelle **12**: 333-351.

Dhingra, A. and P. Chandna (2010). "Hybrid genetic algorithm for SDST flow shop scheduling with due dates: a case study." International Journal of Advanced Operations Management **2**(3): 141-161.

Eberhart, R. C. and J. Kennedy (1995). A new optimizer using particle swarm theory. Proceedings of the sixth international symposium on micro machine and human science, New York, NY.

Ebrahimi, M., S. Fatemi Ghomi and B. Karimi (2014). "Hybrid flow shop scheduling with sequence dependent family setup time and uncertain due dates." Applied Mathematical Modelling **38**(9-10): 2490-2504.

Gowrishankar, K., C. Rajendran and G. Srinivasan (2001). "Flow shop scheduling algorithms for minimizing the completion time variance and the sum of squares of completion time deviations from a common due date." European Journal of Operational Research **132**(3): 643-665.

Grabowski, J. and J. Pempera (2000). "Sequencing of jobs in some production system." European Journal of Operational Research **125**: 535-550.

Grabowski, J. and J. Pempera (2005). "Some local search algorithms for no-wait flow-shop problem with makespan criterion." Computers & Operations Research **32**: 2197–2212.

Gupta, J. N., V. Lauff and F. Werner (2000). On the solution of 2-machine flow shop problems with a common due date. Operations Research Proceedings 1999, Springer.

Hall, N. and C. Sriskandarajah (1996). "A survey of machine scheduling problems with blocking and no-wait in process." Operations Research **44**: 510-525.

Hunsucker, J. and J. Shah (1992). "Performance of Priority Rules in a Due Date Flow Shop." Omega **20**(1): 73-89.

Jolai, F., M. Rabiee and H. Asefi (2012). "A novel hybrid meta-heuristic algorithm for a no-wait flexible flow shop scheduling problem with sequence dependent setup times." International Journal of Production Research **50**(24): 7447 - 7466.

Kennedy, J. and R. C. Eberhart (1997). A discrete binary version of the particle swarm algorithm. Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on, Orlando, FL, IEEE.

King, J. and A. Spachis (1980). "Heuristics for flowshop scheduling." International Journal of Production Research **18**: 343-357.

Liu, B., L. Wang and Y.-H. Jin (2007). "An effective hybrid particle swarm optimization for no-wait flow shop scheduling." International Journal of Advanced Manufacturing Technology **31**: 1001-1011.

Montgomery, D. C. (2008). Design and Analysis of Experiments, John Wiley & Sons.

Nagano, M. S. and D. C. Araújo (2014). "New heuristics for the no-wait flowshop with sequence-dependent setup times problem." Journal of the Brazilian Society of Mechanical Sciences and Engineering **36**(1): 139-151.

Nagano, M. S., A. A. Da Silva and L. A. Nogueira Lorena (2014). "An evolutionary clustering search for the no-wait flow shop problem with sequence dependent setup times." Expert Systems with Applications **41**(8): 3628-3633.

Nagano, M. S., H. H. Miyata and D. C. Araújo (Article in Press). "A constructive heuristic for total flowtime minimization in a no-wait flowshop with sequence-dependent setup times." Journal of Manufacturing Systems.

Nagano, M. S., A. A. d. Silva and L. A. N. Lorena (2012). "A new evolutionary clustering search for a no-wait flowshop problem with set-up times." Engineering Applications of Artificial Intelligence **25**(6): 1114 - 1120.

Panwalkar, S. and C. Koulamas (2012). "An O(n^2) algorithm for the variable common due date, minimal tardy jobs bicriteria two-machine flow shop problem with ordered machines." European Journal of Operational Research **221**(1): 7-13.

Piehler, J. (1960). "Ein beitrag zum reihenfolgeproblem." Unternehmensforschung **4**(3): 138-142.

Pinedo, M. (2002). Scheduling, Theory, Algorithms, and Systems. London, Sydney, Toronto, Mexico, New Delhi, Tokyo, Singapore, Rio de Janeiro, Prentice Hall Inc.

Raaymakers, W. and J. Hoogeveen (2000). "Scheduling multipurpose batch process industries with no-wait restrictions by simulated annealing." European Journal of Operational Research **126**: 131-151.

Rabiee, M., M. Zandieh and A. Jafarian (2012). "Scheduling of a no-wait two-machine flow shop with sequence-dependent setup times and probable rework using robust meta-heuristics." International Journal of Production Research **50**(24): 7428 - 7446.

Rajendran, C. (1994). "A no-wait flowshop scheduling heuristic to minimize makespan." Journal of the Operational Research Society **45**: 472-478.

Reeves, C. (1995). "A genetic algorithm for flowshop sequencing." Computers and Operations Research **22**: 5-13.

Röck, H. (1984). "Some new results in flow shop scheduling." Zeitschrift für Operations Research **28**: 1-16.

Samarghandi, H. (Article in Press). "Studying the effect of server side-constraints on the makespan of the no-wait flow-shop problem with sequence-dependent set-up times." International Journal of Production Research(ahead-of-print): 1-22.

Samarghandi, H. and T. Y. ElMekkawy (2011). "An effective hybrid algorithm for the two-machine no-wait flow shop problem with separable setup times and single server." European Journal of Industrial Engineering **5**(2): 111-131.

Samarghandi, H. and T. Y. ElMekkawy (2012a). "A genetic algorithm and particle swarm optimization for no-wait flow shop problem with separable setup times and makespan criterion." The International Journal of Advanced Manufacturing Technology **61**(9-12): 1101-1114.

Samarghandi, H. and T. Y. ElMekkawy (2012b). "A meta-heuristic approach for solving the no-wait flow-shop problem." International Journal of Production Research **50**(24): 7313-7326.

Samarghandi, H. and T. Y. ElMekkawy (2013). "Two-machine, no-wait job shop problem with separable setup times and single-server constraints." The International Journal of Advanced Manufacturing Technology **65**(1-4): 295-308.

Samarghandi, H. and T. Y. ElMekkawy (2014). "Solving the no-wait flow-shop problem with sequence-dependent set-up times." International Journal of Computer Integrated Manufacturing **27**(3): 213-228.

Sarper, H. (1995). "Minimizing the sum of absolute deviations about a common due date for the two-machine flow shop problem." Applied mathematical modelling **19**(3): 153-161.

Schuster, C. and J. Framinan (2003). "Approximative procedures for no-wait job shop scheduling." Operations Research Letters **31**: 308-318.

Tang, H. B., C. M. Ye and L. F. Jiang (2011). "A New Hybrid Particle Swarm Optimization for Solving Flow Shop Scheduling Problem with Fuzzy Due Date." Advanced Materials Research **189**: 2746-2753.

Tari, F. G. and L. Olfat (2014). "Heuristic rules for tardiness problem in flow shop with intermediate due dates." The International Journal of Advanced Manufacturing Technology **71**(1-4): 381-393.

Tasgetiren, M. F., Y.-C. Liang, M. Sevkli and G. Gencyilmaz (2007). "A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem." European Journal of Operational Research **177**(3): 1930-1947.

Wismer, D. (1972). "Solution of the flowshop-scheduling with no intermediate queues." Operations Research **20**: 689-697.

Ying, K.-C., Z.-J. Lee, C.-C. Lu and S.-W. Lin (2012). "Metaheuristics for scheduling a no-wait flowshop manufacturing cell with sequence-dependent family setups " The International Journal of Advanced Manufacturing Technology **58**(5 - 8): 671 - 682.